

S1C17 16-bit Microcontroller Introduction

- S1C17 Roadmap
- S1C17 Core Facts
- S1C17701 Details
- S1C17 Development Tools

S1C17 16-bit Microcontroller Roadmap

S1C17 Family Roadmap

**LCD controller
I/O controller**
Healthcare
Home appliance
Operation panel
Metering

*High-Speed
48MHz
5V I/O
USB*

09/2007 ES

S1C17801
CSTN LCDC
128KB Flash
4KB/6KB RAM
10bit ADC
USB

10/2007 ES

S1C17501
128KB Flash
4KB/6KB RAM
10bit ADC
USB

S1C17802
CSTN LCDC
256KB Flash
4KB/6KB RAM
10bit ADC
USB
Motor controller

S1C17502
256KB Flash
4KB/6KB RAM
10bit ADC
USB
Motor controller

Dot Matrix Dr.
Sports Watch
Remote Cont.
Thermostat
Education Tool
IP Phone
LCD Game

08/2007 ES

S1C17701
56x32 4G/S LCD
64KB Flash
4KB RAM
IR Controller

12/2007 ES

S1C17702
72x32 4G/S LCD
128KB Flash
8KB RAM
Wibree I/F
Remote Controller
ADPCM Decoder

S1C17703
128x32 4G/S LCD
256KB Flash
8KB RAM
Wibree I/F
Remote Controller
ADPCM Decoder

*Low Power
HALT<2uA*

Segment Dr.
Sports watch
Healthcare
Home appliance
Operation panel
Metering

S1C17601
40x4 LCD Driver
Digital Contr. Osc.
32KB Flash
4KB RAM
10bit ADC

S1C17602
48x4 LCD Driver
Digital Contr. Osc.
64KB Flash ROM
8KB RAM
10bit ADC

S1C17603
64x8 LCD Driver
Digital Contr. Osc.
256KB Flash ROM
16KB RAM
10bit ADC

2007

2008

2009

S1C17

Core Facts

What is S1C17?

Epson has developed a **new 16-bit RISC architecture** for next-generation microcontrollers targeted at embedded display and control applications.

The S1C17 family will be a long-term replacement for the existing Epson S1C88 8-bit microcontrollers, offering the same **outstanding features** like **high-resolution dot matrix** and **segment-type LCD drivers**, **integrated charge pumps** and **low power consumption**.

To better support our customers need for production flexibility and easy development, the microcontrollers of S1C17 family will mostly be delivered as **flash types**.

- Linear address space
 - Instruction/data/IO can access the **same 16MB linear memory**
 - No segments / banks required**
- GNU C compiler supports 16MB space
 - 3 memory models to optimize code density
 - Small → up to 64kB addressable (16bit pointer)
 - Middle → up to 1MB addressable (20bit pointer)
 - Regular → up to 16MB addressable (24bit pointer)
- RISC instruction set optimized for C language
- **In-Circuit-Debugging function included**
- CPU core **gate count smaller than other 8-bit CPU cores**
- High performance execution similar to 32-bit RISC cores
 - 3-stage pipeline: Fetch-Decode-Execute → 1 CLK/instruction
 - High speed interrupt: 3 CLK (includes vector table fetch)
 - Up to 90MHz core speed**

Processor type

- Seiko Epson original 16-bit RISC processor
- 0.35–0.15 μm low power CMOS process technology

Operating-clock frequency

- 90 MHz maximum (depending on the processor model and process technology)

Instruction set

- Code length: 16-bit fixed length
- Number of instructions: 111 basic instructions (184 including variations)
- Execution cycle: Main instructions executed in one cycles
- Extended immediate instructions: Immediate extended up to 24 bits
- Compact and fast instruction set optimized for development in C language

Register set

- Eight 24-bit general-purpose registers
- Two 24-bit special registers
- One 8-bit special register

Memory space and bus

- Up to 16M bytes of memory space (24-bit address)
- Harvard architecture using separated instruction bus (16 bits) and data bus (32 bits)

Interrupts

- Reset, NMI, and 32 external interrupts supported
- Address misaligned interrupt
- Debug interrupt
- Direct branching from vector table to interrupt handler routine
- Programmable software interrupts with a vector number specified (all vector numbers specifiable)

Power saving

- HALT (halt instruction)
- SLEEP (slp instruction)

Coprocessor interface

- ALU instructions can be enhanced

S1C17 Explained: Registers

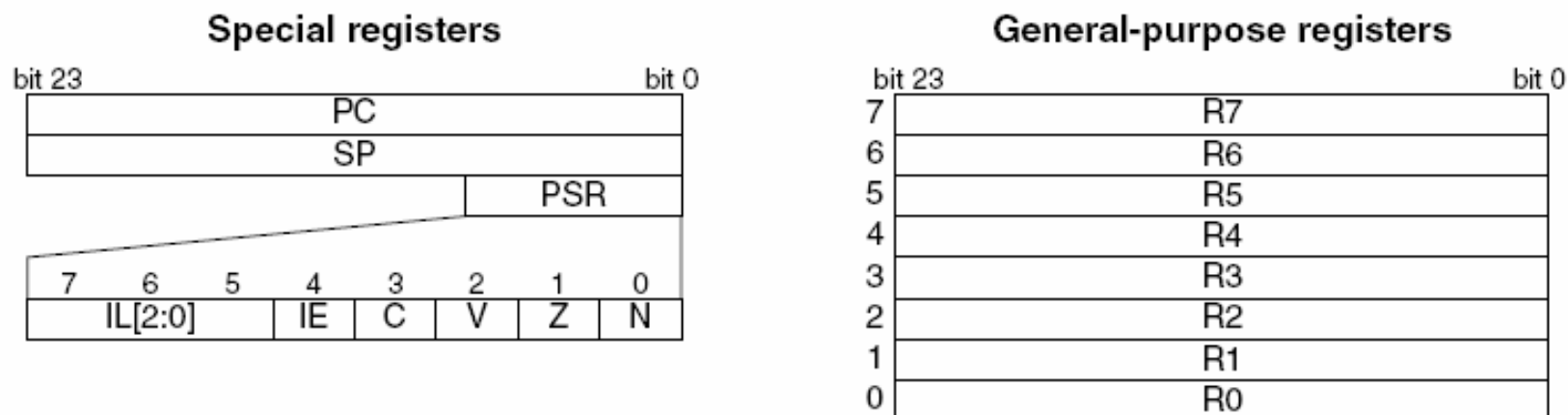


Figure 2.1 Registers

The eight registers R0–R7 (r0–r7) are 24-bit general-purpose registers that can be used for data manipulation, data transfer, memory addressing, or other general purposes. The contents of all of these registers are handled as 24-bit data or addresses. 8- or 16-bit data can be sign- or zero-extended to a 24-bit quantity when it is loaded into one of these registers using a load instruction or a conversion instruction. When these registers are used for address references, 24-bit memory space can be accessed directly.

At initial reset, the contents of the general-purpose registers are set to 0 (may be indeterminate without initialization depending on the configuration).

Register Operations

The S1C17 Core can handle 8-, 16-, and 24-bit data in register operations. In this manual, data sizes are expressed as follows:

8-bit data	Byte, B, or b
16-bit data	Word, W, or w
24-bit data	Address data, A, a

Memory Operations

The S1C17 Core can handle 8-, 16-, and 32-bit data in memory operations. In this manual, data sizes are expressed as follows:

8-bit data	Byte, B, or b
16-bit data	Word, W, or w
32-bit data	Address data, A, a

S1C17 Explained: Address Map

The S1C17 Core supports a 24-bit address allowing linear use of address space up to 16M bytes. Addresses 0xffff00 to 0xfffff are reserved as an I/O area for the core. In addition to this area, a 64-byte area located in the user RAM is required for debugging.

Figure 4.1.1 shows the address space of the S1C17 Core.

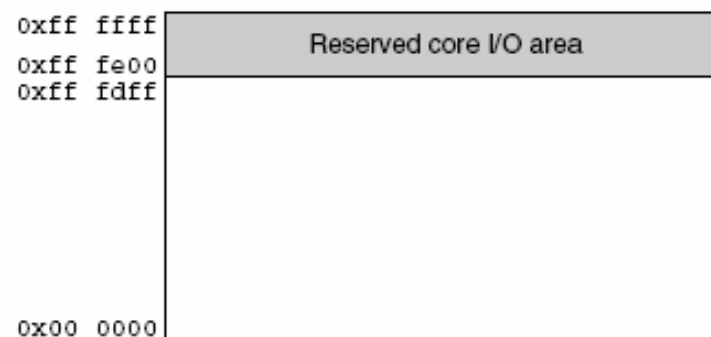


Figure 4.1.1 Address Space of the S1C17 Core

Following initial reset, the processor loads the reset vector (address of the reset handler routine) into the PC and starts executing instructions beginning with the address. As the instructions in the S1C17 Core are fixed to 16 bits in length, the PC is incremented by 2 each time an instruction is fetched from the address indicated by the PC. In this way, instructions are executed successively.

When a branch instruction is executed, the processor checks the PSR flags and whether the branch conditions have been satisfied, and loads the jump address into the PC.

When an interrupt occurs, the processor loads the address for the interrupt handler routine from the vector table into the PC.

The vector table contains interrupt vectors beginning with the reset vector and is located from the address set in the TTBR register (0xffff80). The start address can be set to the TTBR in the configuration.

4.2.1 Trap Table Base Register (TTBR, 0xffff80)

Register name	Address	Bit	Name	Function	Setting	Init.	R/W	Remarks
Trap table base register	FFFF80 (L)	D31–24	–	Unused (fixed at 0)	0x0	0x0	R	
		D23	TTBR23	Trap table base address TTBR[7:0] is fixed at 0x0.	0x0–0xFFFD00 (256 byte units)	*	R	Initial value is set by the TTBR pins of the C17 macro.
		D0	TTBR0					

This is a read-only register that contains the trap table base address.

The trap table (also called a vector table) contains the vectors to the interrupt handler routines (handler routine start address) that will be read by the S1C17 Core to execute the handler when an interrupt occurs. The boot address from which the program starts running after a reset must be written to the top of the trap table.

S1C17 Explained: Instruction Pipeline

Basic instruction stages

Instruction fetch	Instruction decode	Instruction execution / Memory access / Register write
-------------------	--------------------	--

Hereinafter, each stage is represented by the following symbols:

F (for Fetch): Instruction fetch

D (for Decode): Instruction decode

E (for Execute): Instruction execution, memory access, register write

Pipelined operation

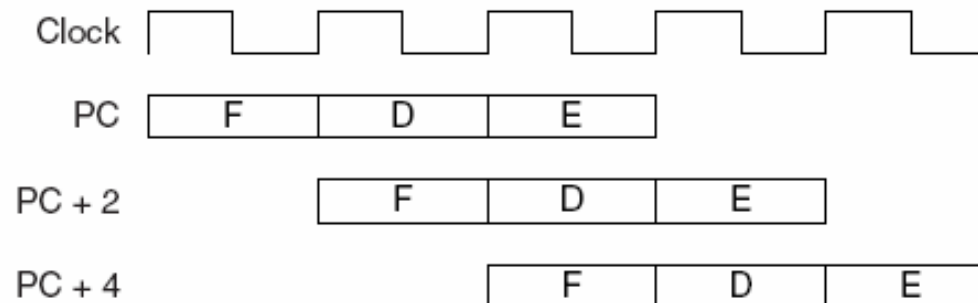


Figure 6.2.1.1 Pipelined Operation

S1C17 Explained: Interrupts

Table 6.3.1.1 Vector Address and Priority of Interrupts

Interrupt	Vector address (Hex)	Priority
Reset	TTBR + 0x00	High ↑ ↓ Low
Address misaligned interrupt	TTBR + 0x04	
Debug interrupt	(0xfffc00)	
NMI	TTBR + 0x08	
Software interrupt	TTBR + 0x00 to TTBR + 0x7c	Low
Maskable external interrupt	TTBR + 0x00 to TTBR + 0x7c	

When two or more interrupts occur simultaneously, they are processed in order of priority beginning with the one that has the highest priority.

When an interrupt occurs, the processor disables interrupts that would occur thereafter and performs interrupt handling. To support multiple interrupts (or another interrupt from within an interrupt), set the IE flag in the PSR to 1 in the interrupt handler routine to enable interrupts during interrupt handling. Basically, even when multiple interrupts are enabled, interrupts whose priorities are below the one set by the IL[2:0] bits in the PSR are not accepted.

Table 6.3.2.1 Vector List

Vector No. Software interrupt No.	Interrupt	Vector address
0 (0x00)	Reset	TTBR + 0x00
1 (0x01)	Address misaligned interrupt	TTBR + 0x04
2 (0x02)	NMI	TTBR + 0x08
3 (0x03)	Maskable external interrupt 3	TTBR + 0x0c
:	:	:
31 (0x1f)	Maskable external interrupt 31	TTBR + 0x7c

S1C17 Explained: Power States

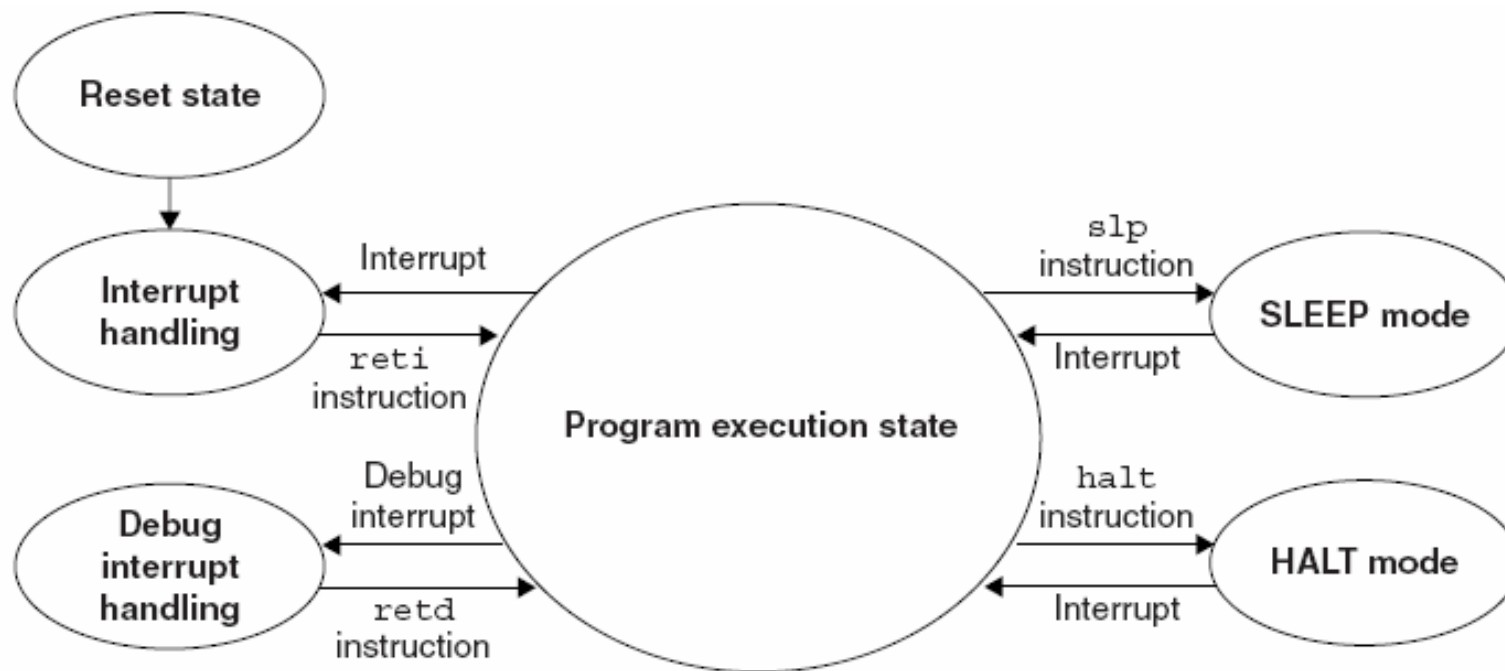


Figure 6.1.1 Processor Status Transition Diagram

HALT mode

Program execution is halted at the same time that the S1C17 Core executes the `halt` instruction, and the processor enters HALT mode.

HALT mode commonly turns off only the S1C17 Core operation, note, however that modules to be turned off depend on the implementation of the clock control circuit outside the core. Refer to the technical manual of each model for details.

SLEEP mode

Program execution is halted at the same time the S1C17 Core executes the `slp` instruction, and the processor enters SLEEP mode.

SLEEP mode commonly turns off the S1C17 Core and on-chip peripheral circuit operations, thereby it significantly reduces the current consumption in comparison to HALT mode. However, modules to be turned off depend on the implementation of the clock control circuit outside the core. Refer to the technical manual of each model for details.

- **Instruction break**

A debug interrupt is generated before the set instruction address is executed. An instruction break can be set at two addresses.

- **Single step**

A debug interrupt is generated every instruction executed.

- **Forcible break**

A debug interrupt is generated by an external input signal.

- **Software break**

A debug interrupt is generated when the `brk` instruction is executed.

When a debug interrupt occurs, the processor performs the following processing:

(1) Suspends the instructions currently being executed.

(2) Saves the contents of the PC and PSR, and R0, in that order, to the addresses specified below.

PC/PSR → DBRAM + 0x0

R0 → DBRAM + 0x4 (DBRAM: Start address of the work area for debugging in the user RAM)

(3) Loads address 0xfffc00 to PC and branches to the debug interrupt handler routine.

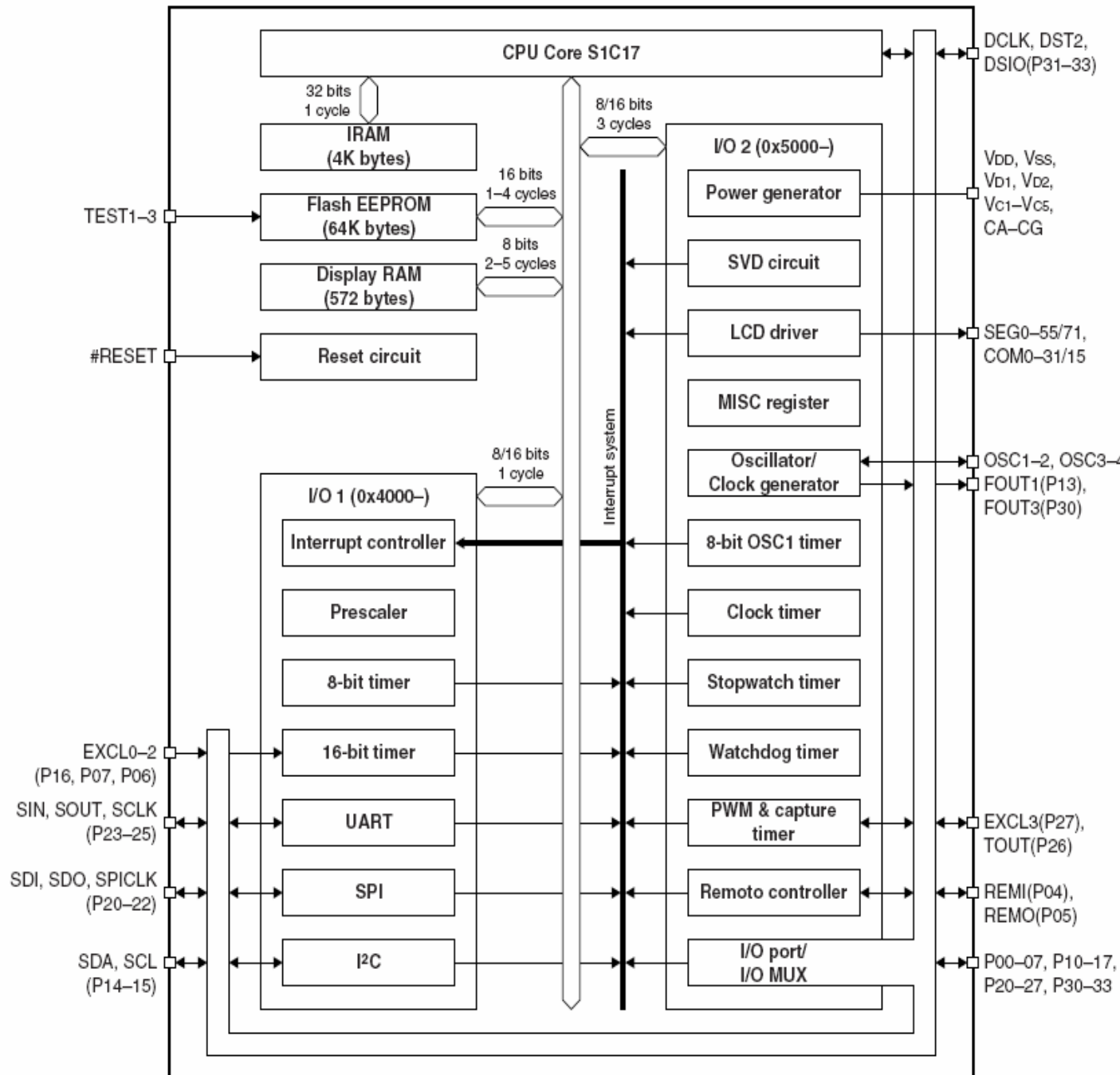
In the interrupt handler routine, the `ret d` instruction should be executed at the end of processing to return to the suspended instructions. When returning from the interrupt by the `ret d` instruction, the processor restores the saved data in order of the R0 and the PC and PSR.

Neither hardware interrupts nor NMI interrupts are accepted during a debug interrupt.

S1C17701

- **Low power flash MCU**
 - Operating voltage: 1.8V
 - Current consumption: Sleep/1uA, Halt/2uA
 - Current consumption equivalent to Mask ROM products
- **MCU + dot matrix LCD driver + flash memory**
 - LCD driver: 56seg x 32com
 - Built-in booster power supply circuit
 - 64KB flash ROM
 - 4KB RAM
- **Built-in remote control circuit**

S1C17701: Block Diagram



- **64KB Flash memory**
- **4KB RAM**
- **VDD=1.8 .. 3.6V**
- **Flash self-writing supported**
- **56x32 LCD driver**
- **4 wire interface for on-chip debugging and flash writing**
- **TQFP24-144 package**
- **BGA package under development**

S1C17701: Memory Map

		Peripheral function	(Device size)
0xff ffff	Reserved for core I/O area (1K bytes, 1 cycle)	0x5360-0x53ff reserved	—
0xff fc00		0x5340-0x535f Remote controller	(8 bits)
0xff fbff	reserved	0x5320-0x533f MISC registers	(8 bits)
0x08 0560		0x5300-0x531f PWM & capture timer	(16 bits)
0x08 055f	Display RAM area (572 bytes, 2-5 cycles) (Device size: 8 bits)	0x52c0-0x52ff reserved	—
		0x52a0-0x52bf Port MUX	(8 bits)
0x08 0000		0x5280-0x529f reserved	—
0x07 ffff	reserved	0x5200-0x527f P ports	(8 bits)
0x01 8000		0x5140-0x51ff reserved	—
0x01 7fff		0x5120-0x513f Power controller	(8 bits)
		0x5100-0x511f SVD circuit	(8 bits)
	Flash area (64K bytes, 1-4 cycles) (Device size: 16 bits)	0x50e0-0x50ff reserved	(8 bits)
		0x50c0-0x50df 8-bit OSC1 timer	(8 bits)
		0x50a0-0x50bf LCD driver	(8 bits)
		0x5080-0x509f Clock generator	(8 bits)
		0x5060-0x507f Oscillator	(8 bits)
0x00 8000	Trap table	0x5040-0x505f Watchdog timer	(8 bits)
0x00 7fff	reserved	0x5020-0x503f Stopwatch timer	(8 bits)
0x00 6000		0x5000-0x501f Clock timer	(8 bits)
0x00 5fff	Internal peripheral area 2 (1K bytes, 3 cycles)		
0x00 5000		0x4360-0x43ff reserved	(16 bits)
0x00 4fff	reserved	0x4340-0x435f I ² C	(16 bits)
0x00 4400		0x4320-0x433f SPI	(16 bits)
0x00 43ff	Internal peripheral area 1 (1K bytes, 1 cycle)	0x42e0-0x431f Interrupt controller	(16 bits)
		0x4280-0x42df reserved	(16 bits)
0x00 4000		0x4260-0x427f 16-bit timer Ch. 2	(16 bits)
0x00 3fff	reserved	0x4240-0x425f 16-bit timer Ch. 1	(16 bits)
0x00 1000		0x4220-0x423f 16-bit timer Ch. 0	(16 bits)
0x00 0fff	Debug RAM area (64 bytes)	0x4200-0x421f 8-bit timer	(16 bits)
0x00 0fc0		0x4120-0x41ff reserved	(8 bits)
	IRAM area (4K bytes, 1 cycle) (Device size: 32 bits)	0x4100-0x411f UART	(8 bits)
		0x4040-0x40ff reserved	(8 bits)
		0x4020-0x403f Prescaler	(8 bits)
0x00 0000		0x4000-0x401f reserved	(8 bits)

Figure 3.1 S1C17701 Memory Map

S1C17701: Power Supply Circuit

The operating voltage range of the S1C17701 is as follows:
For normal operation: 1.8 V to 3.6 V
For Flash programming: 2.7 V to 3.6 V

Table 4.2.1 Power Supply Circuit

Circuit	Power supply circuit	Output voltage
Oscillator and internal logic circuits	Internal logic voltage regulator	V _{D1}
LCD system voltage regulator	Power voltage booster	V _{DD} or V _{D2}
LCD driver	LCD system voltage regulator	V _{C1} to V _{C5}

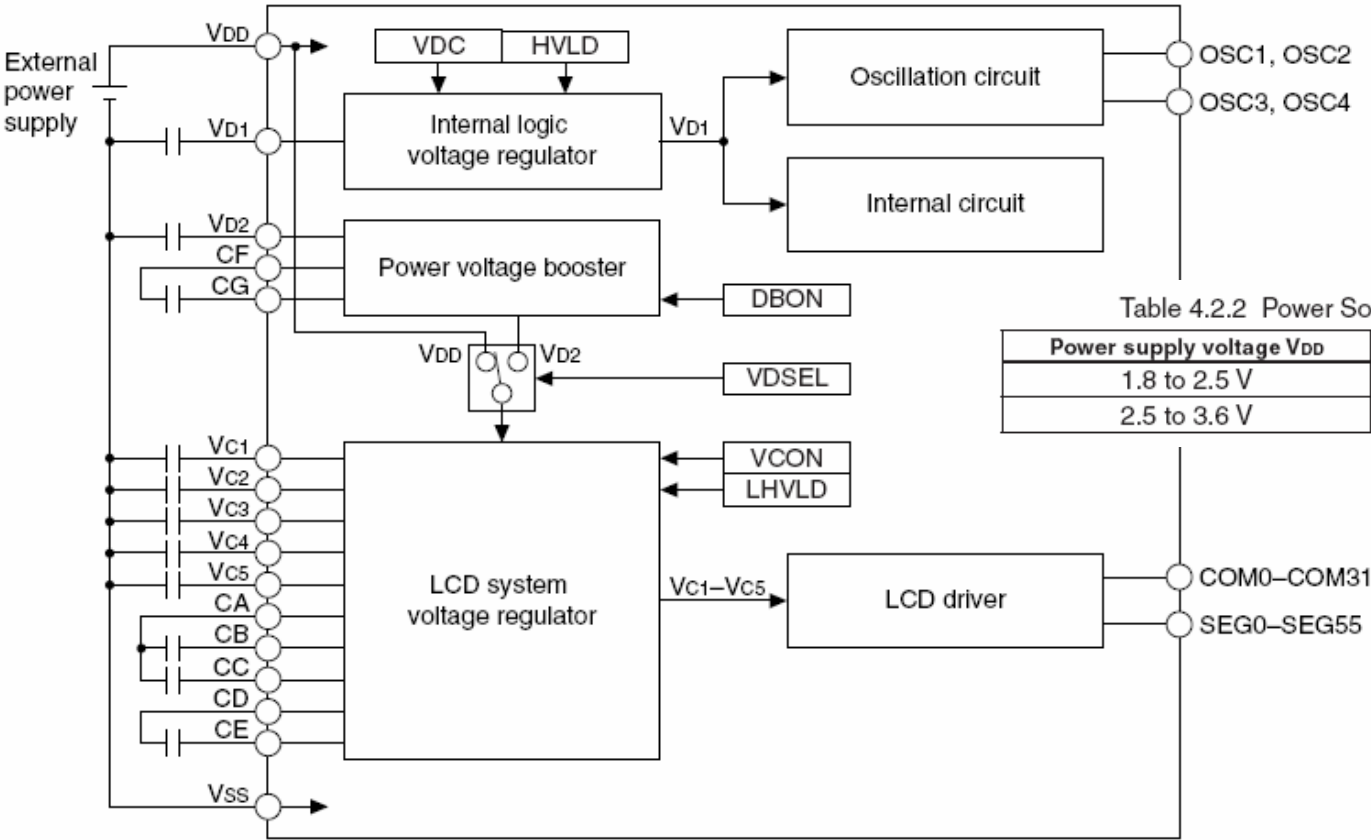


Table 4.2.2 Power Source for LCD System Voltage Regulator

Power supply voltage V _{DD}	Power source for the LCD system voltage regulator
1.8 to 2.5 V	V _{D2} ($\cong 2 \cdot V_{DD}$)
2.5 to 3.6 V	V _{DD}

Figure 4.2.1 Configuration of Power Supply Circuit

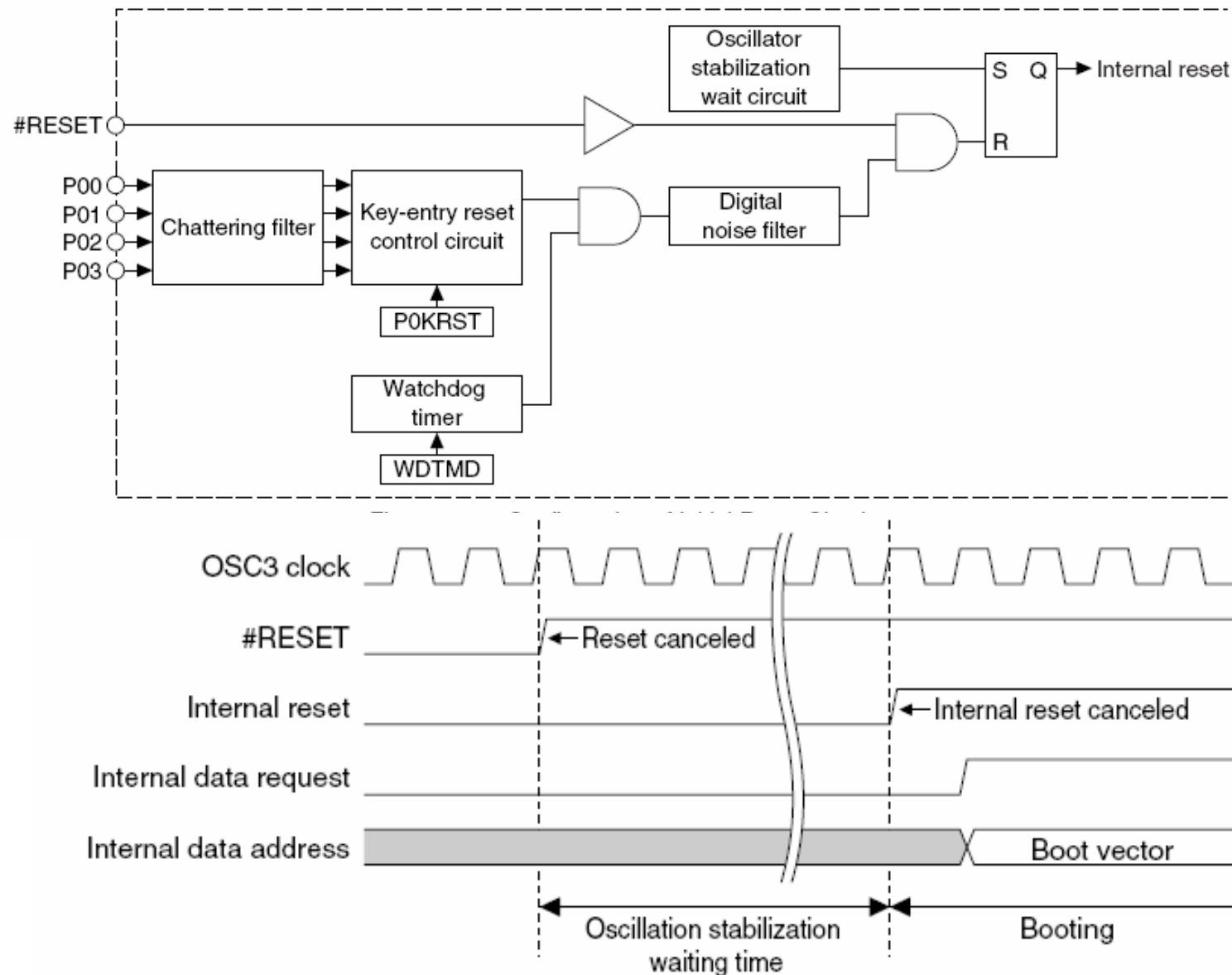


Figure 5.2.1 Operation Sequence Following Cancellation of Initial Reset

The S1C17701 contains 28 I/O ports (P0[7:0], P1[7:0], P2[7:0], and P3[3:0]) that can be directed for input or output with software. Although the I/O pins, except for some I/O ports, are shared with internal peripheral modules, the pins can be used as general-purpose input/output ports unless they are used for the peripheral modules.

Figure 10.1.1 shows the structure of a typical I/O port.

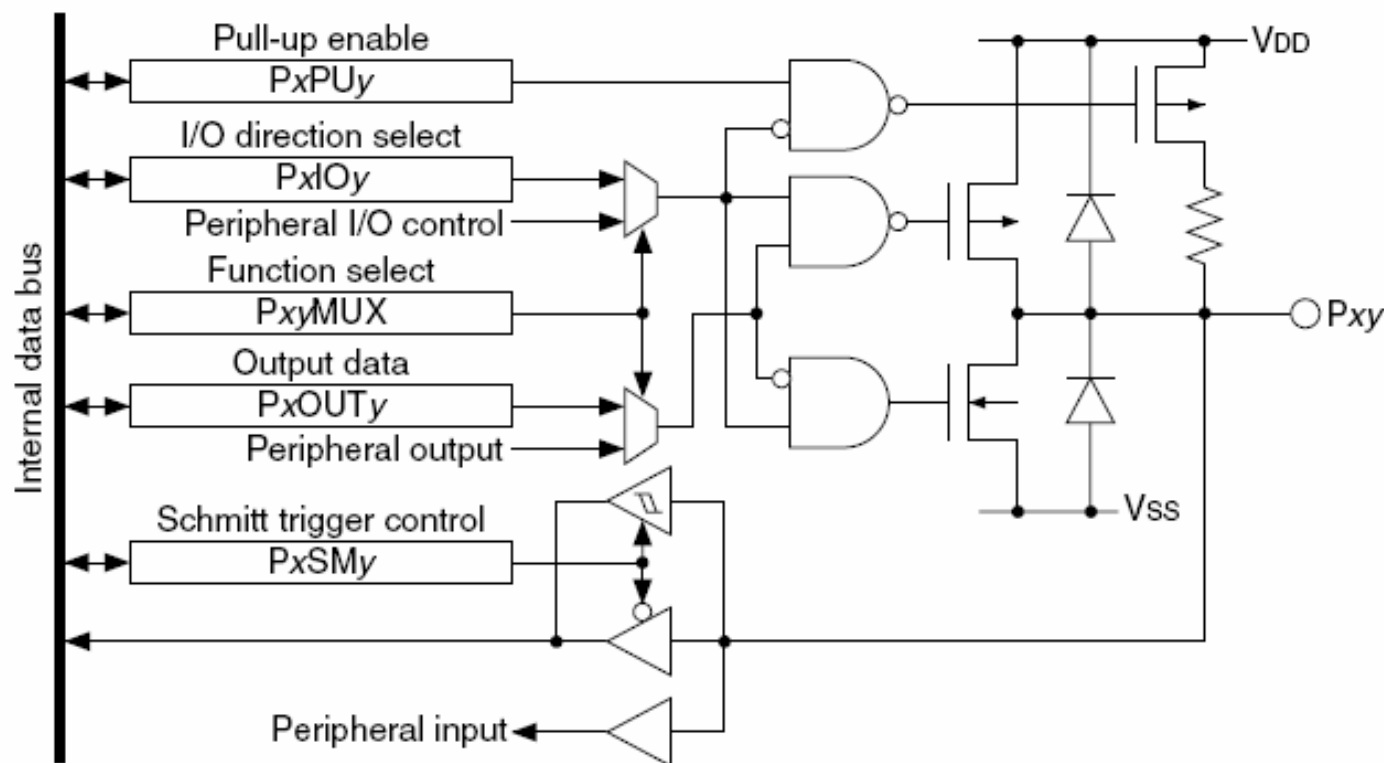


Figure 10.1.1 Structure of I/O Port

The P0 and P1 ports can generate input interrupts.

The P0[3:0] ports can be used as the key-entry reset ports (see Section 5.1.2, “P0 Port Key-Entry Reset,” for details).

The S1C17701 equipped with one channel of UART. The UART performs asynchronous data transfer from/to an external serial device in a 150 to 115200 bps transfer rate. The UART contains two-byte receive data buffer and one-byte transmit data buffer allowing full-duplex communication. The transfer clock is internally generated using a timer module or an external clock is input from the SCLK (P25) pin. The character length (seven or eight bits), number of stop bits (one or two bits), and parity mode (even, odd, or none) are programmable. The start bit is fixed at one bit. In data receive operation, overrun, framing, and parity errors are detectable. The UART can generate three types of interrupts (transmit buffer empty, receive buffer full, and receive error), this makes it possible to process serial data transfer simply in an interrupt handler.

Furthermore, the UART module contains an RZI modulator/demodulator, allowing an infrared-ray communication circuit to be configured based on IrDA 1.0 simply by adding an external circuit.

Figure 18.1.1 shows the structure of the UART.

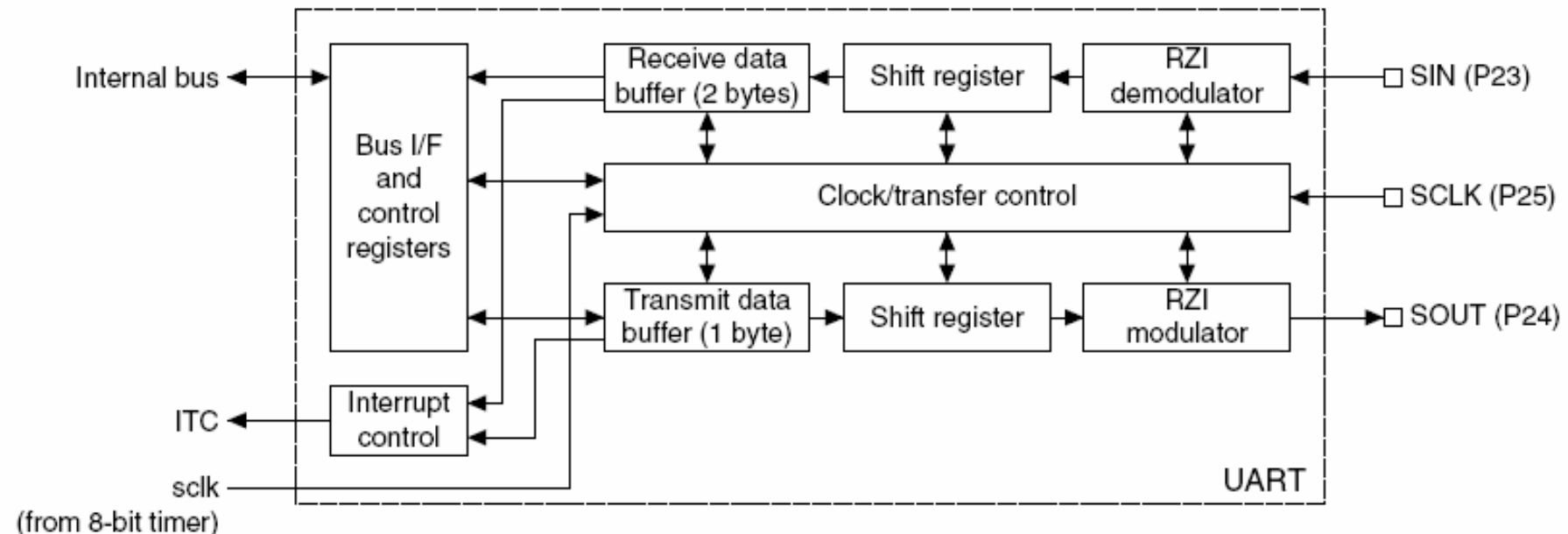


Figure 18.1.1 Structure of UART

The S1C17701 equipped with a synchronous serial interface module (hereafter SPI). The SPI module supports both master and slave modes and performs 8-bit serial data transfer. Data transfer timing (clock phase and polarity variations) is selectable from among 4 types.

The SPI includes a transmit data buffer and a receive data buffer separately from the shift registers, and can generate two types of interrupts (transmit data buffer empty and receive data buffer full), this makes it possible to process continuous serial data transfers simply in an interrupt handler.

Figure 19.1.1 shows the structure of the SPI module.

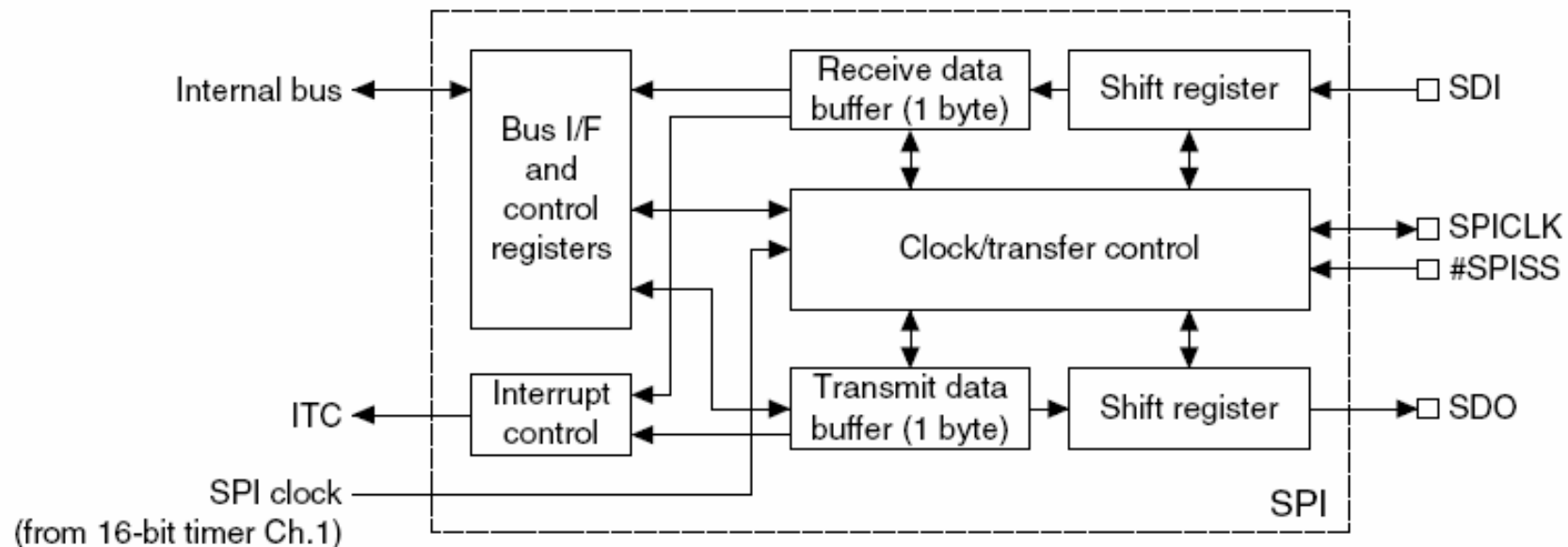


Figure 19.1.1 Structure of SPI Module

The S1C17701 equipped with an I²C bus interface module for high-speed synchronous serial communication. This I²C module operates as a master using the clock supplied from the 16-bit timer Ch.2. It supports standard (100 kbps) and fast (400 kbps) modes, and 7-bit/10-bit slave addressing. The I²C module includes a clock stretch and noise remove functions to secure reliable data transfer.

Also it can generate two types of interrupts (transmit and receive interrupts), this makes it possible to process continuous serial data transfer simply in an interrupt handler.

Figure 20.1.1 shows the structure of the I²C module.

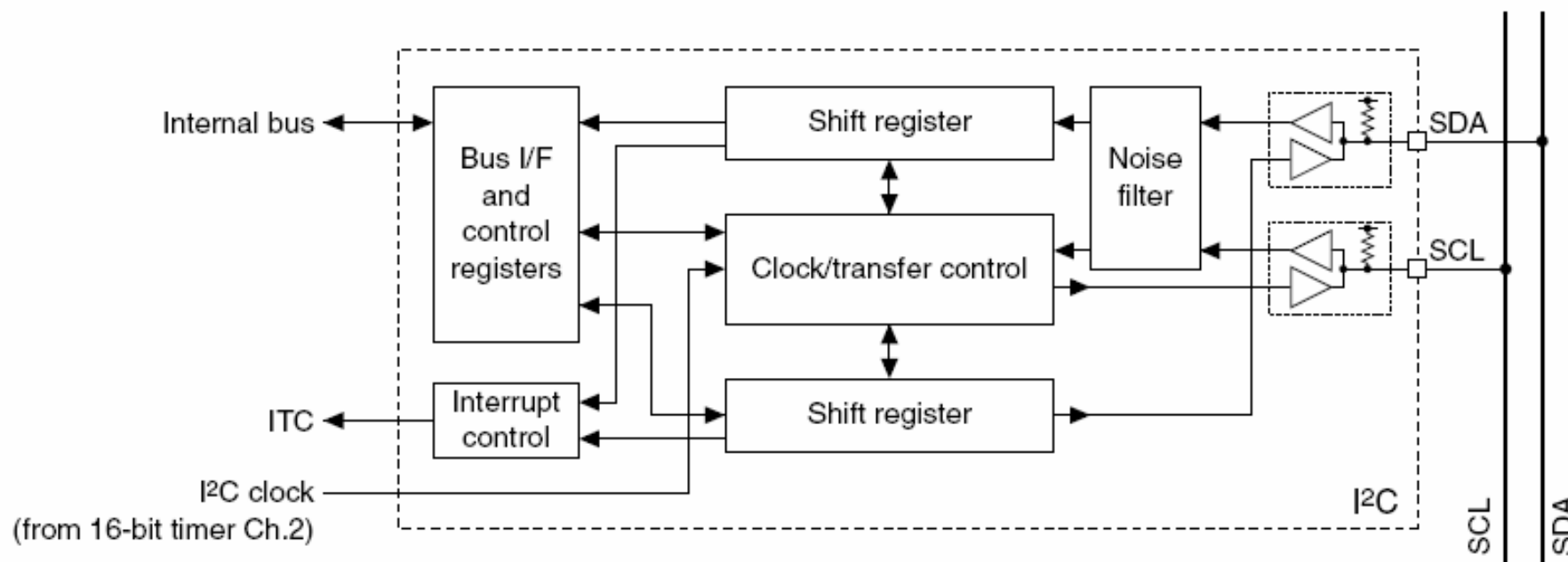


Figure 20.1.1 Structure of I²C Module

S1C17701: LCD Driver

The S1C17701 is equipped with an LCD driver which can drive an LCD panel with a maximum of 1,792 pixels (56 segments × 32 commons).

Figure 22.1.1 shows the configuration of the LCD driver and the drive power supply.

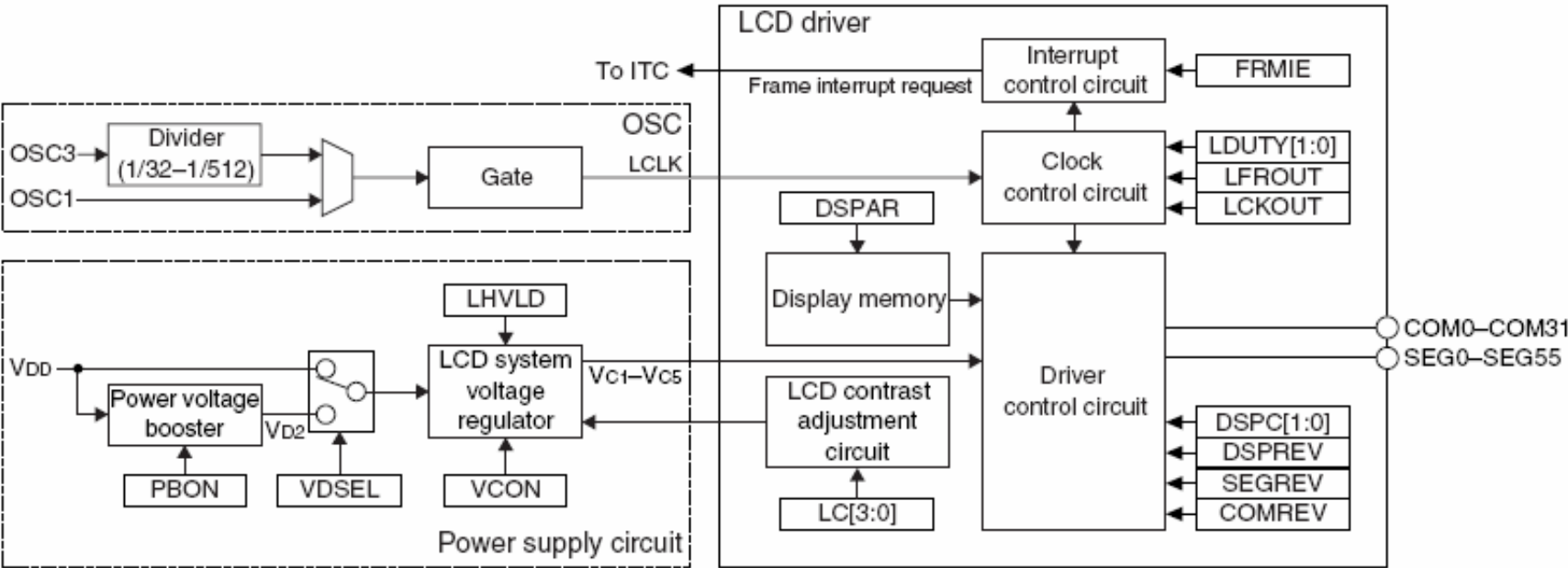


Figure 22.1.1 Configuration of LCD Driver and Drive Power Supply

Table 22.4.1 Setting the Drive Duty

LDUTY[1:0]	Duty	Effective COM pins	Effective SEG pins	Max number of pixels
0x3	Reserved	—	—	—
0x2	1/32	COM0—COM31	SEG0—SEG55	1,792 pixels
0x1	1/16	COM0—COM15	SEG0—SEG71	1,152 pixels
0x0	Reserved	—	—	—

(Default: 0x0)

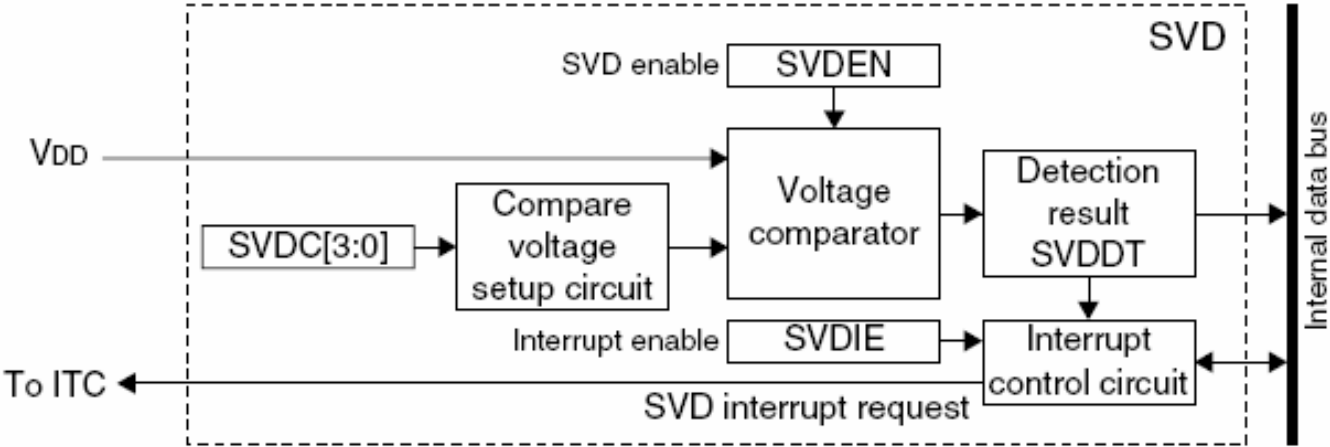


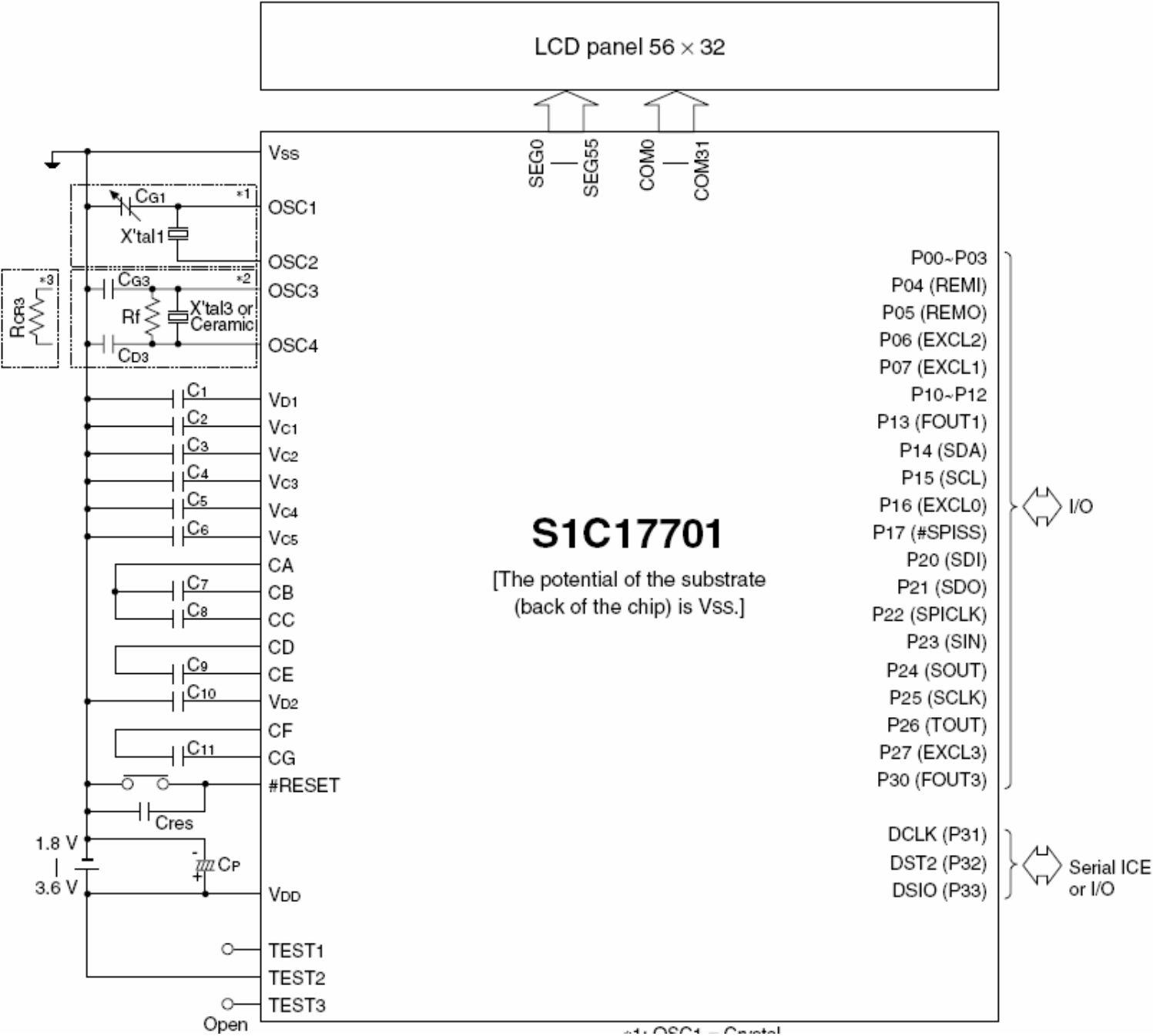
Figure 23.1.1 Structure of SVD Module

Table 23.2.1 Setting the Compare Voltage

SVDC[3:0]	Compare voltage
0xf	2.7 V
0xe	2.6 V
0xd	2.5 V
0xc	2.4 V
0xb	2.3 V
0xa	2.2 V
0x9	2.1 V
0x8	2.05 V
0x7	2.0 V
0x6	1.95 V
0x5	1.9 V
0x4	1.85 V
0x3	1.8 V
0x2 to 0x0	Reserved

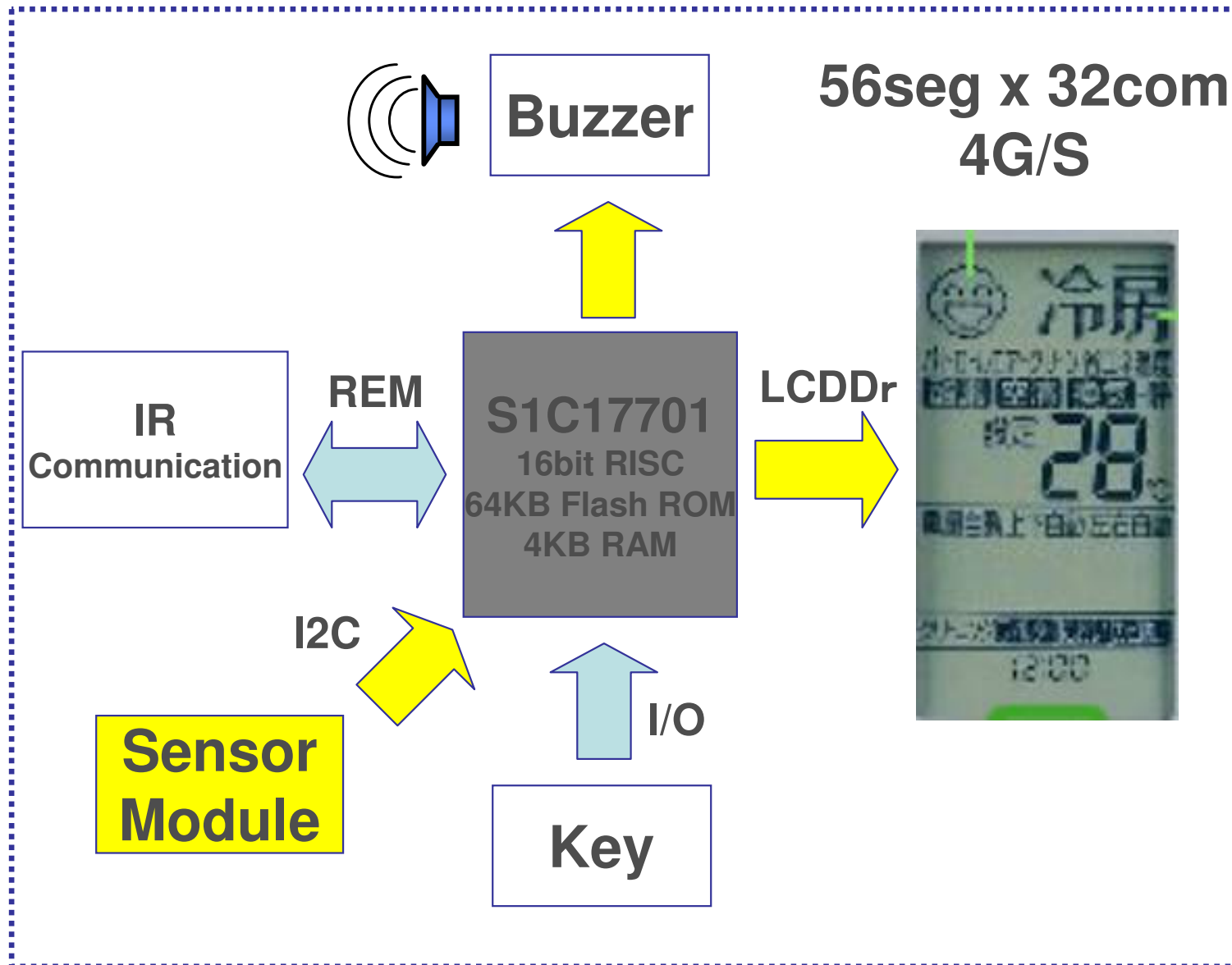
(Default: 0x0)

S1C17701: External Wiring



- ◆ Customers who want to **enhance displays** in their current products or already use a monochrome LCD
- ◆ Customers who are **developing battery-driven products**
- ◆ Customers who need to use **space-saving single-chip MCU + LCD driver**
- ◆ **Applications using currently S1C8F360**
S1C17701 features higher performance, easier development, faster flash-writing

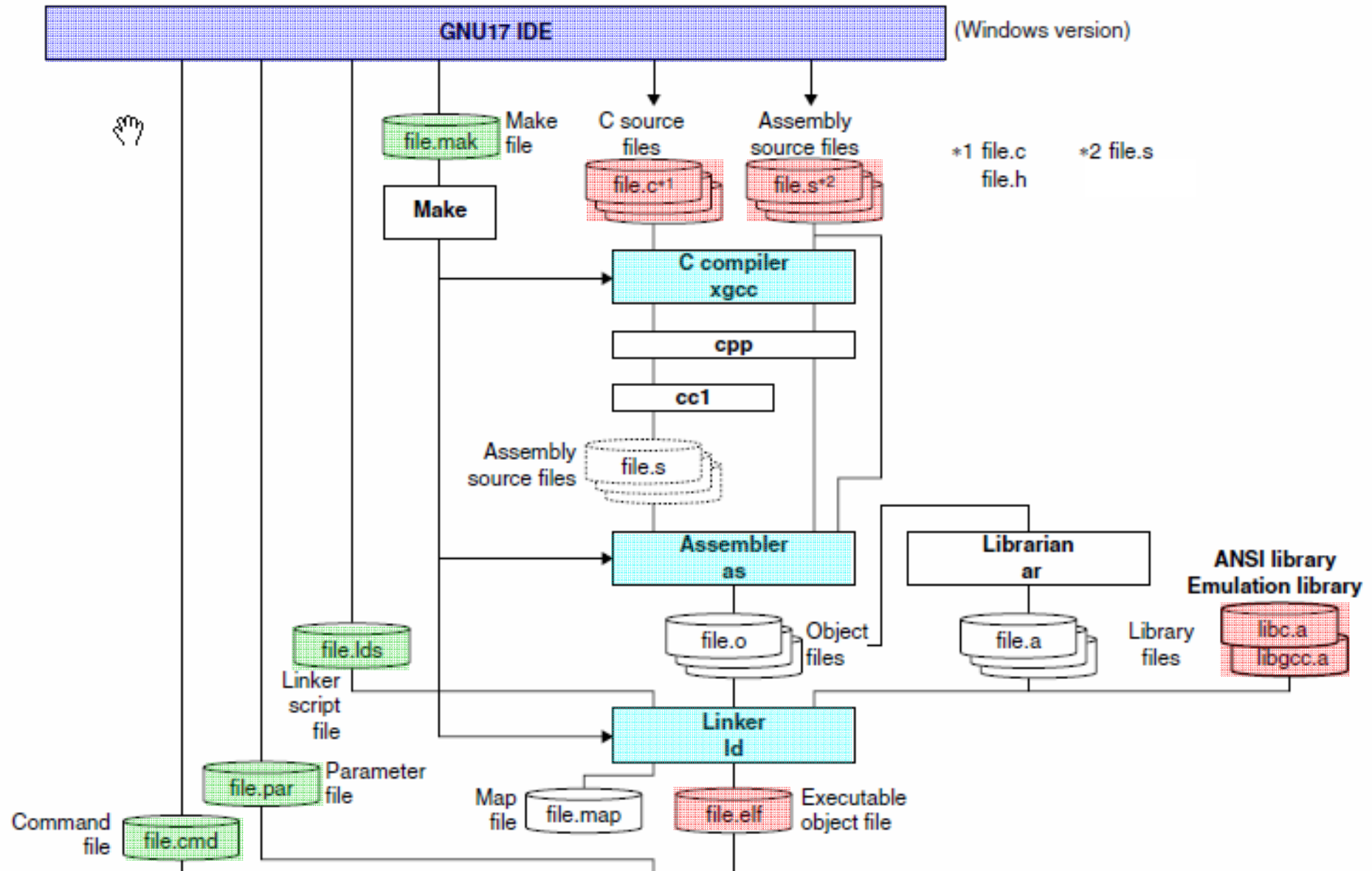
Remote Control Example



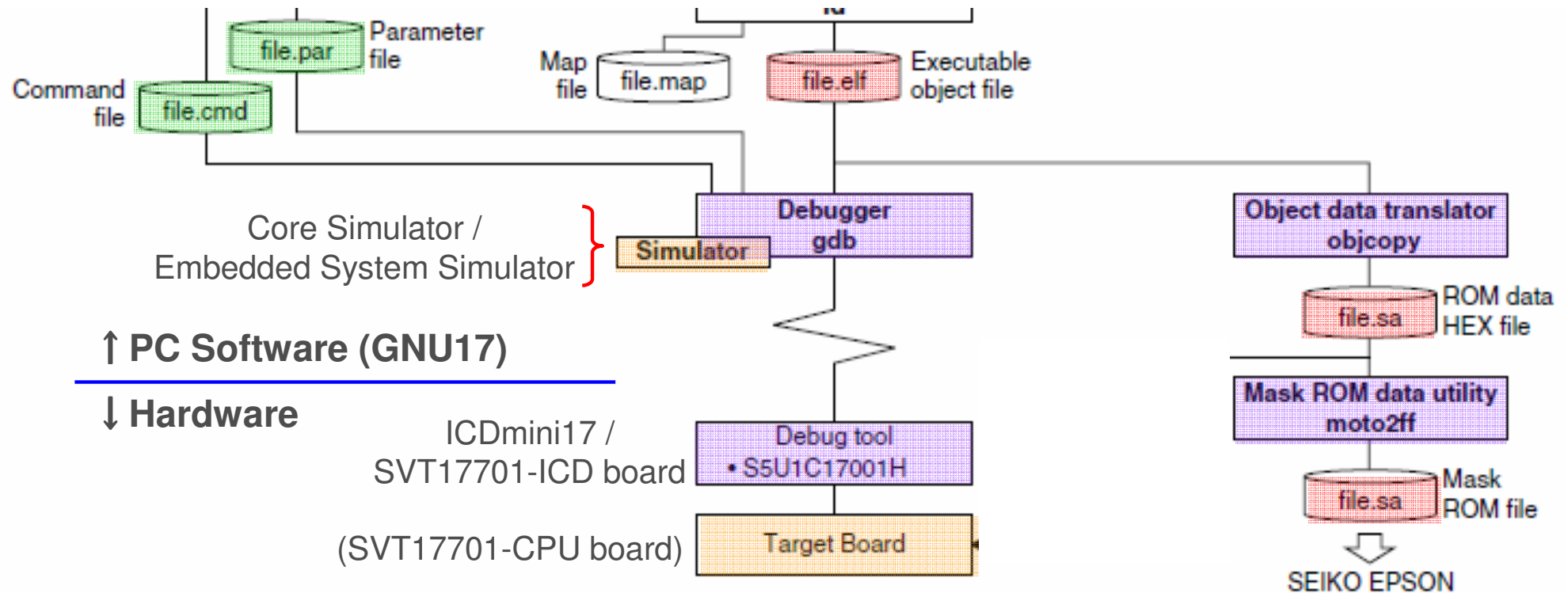
S1C17 Software Development Environment

- Overview
- GNU17v.1.0 - Software Tools -
 - IDE
 - Tool Chain (C compiler, libraries, Assembler, Linker, etc.)
 - GDB debugger
 - Toppers/osek
 - Simulator
 - Sample programs
- ICDmini17 v.1.0 - In Circuit Debugger -
- SVT17701 - Simple Software Evaluation Tool –
- Relation between GNU17 and SVT17701
- Comparison with C33, C88 development tools
- Release Schedule

Overview (1)

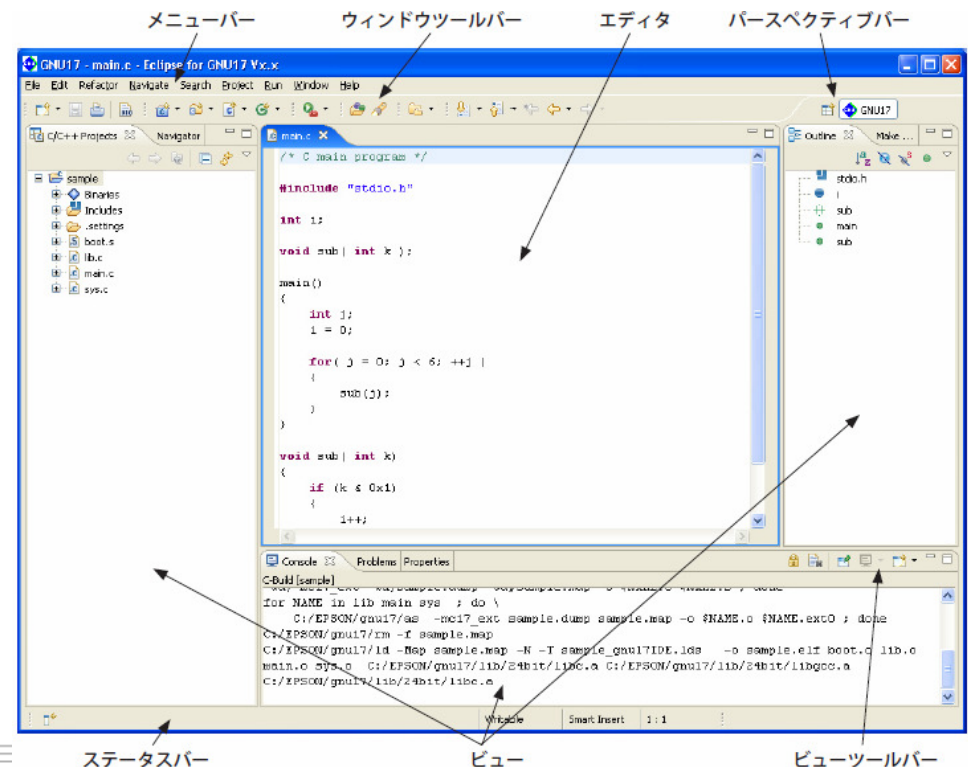


Overview (2)



GNU17v.1.0 (1) IDE

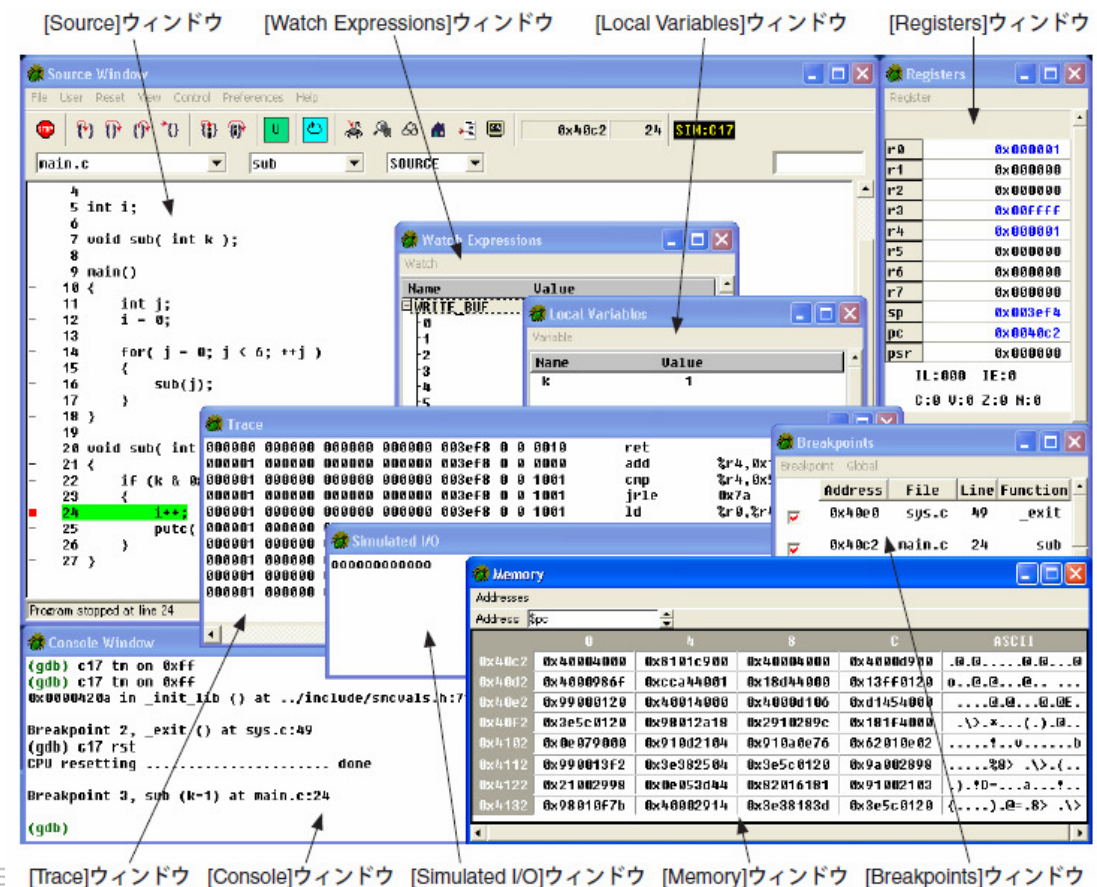
- Integrated Development Environment based on the Eclipse platform which is popular as open source IDE.
- Same look & feel with GNU33.
- Automatically make compile & debug definition files according with user definition.
...make file, linker script file, debugger command file, par file...
- Edit Source code, launch debugger.



- Based on the GNU GCC ver.3.3.2
- Same base with GNU33
- Support 3 memory model to optimize code size.
 - Small 64kB max. 16bit pointer
 - Middle 1MB max. 20bit addressing
 - Regular 16MB max. 24bit addressing
- 2 pass make assembler to minimize extension instruction (EXT) number.
- Special printf() / scanf() sample without emulation library to reduce code size.

GNU17v.1.0 (3) Debugger

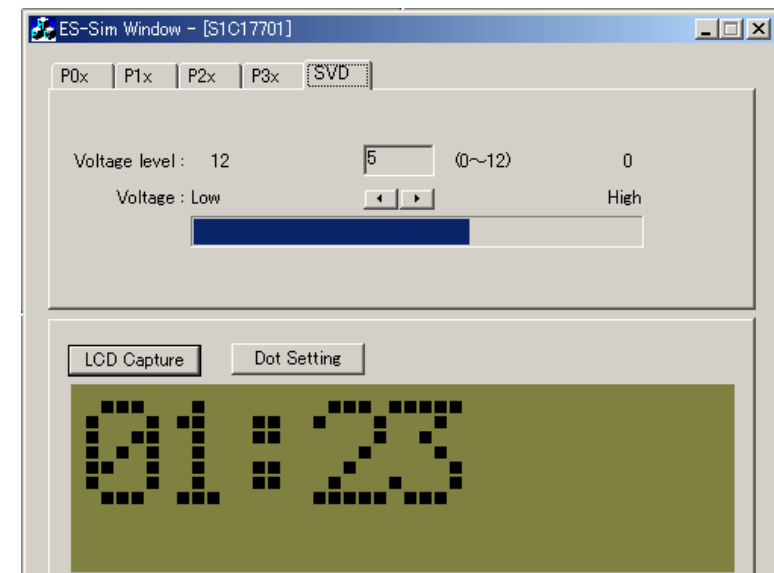
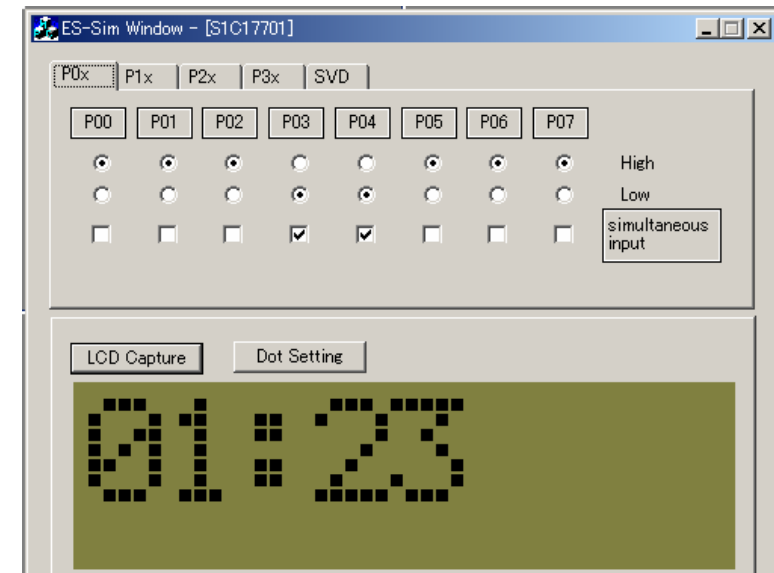
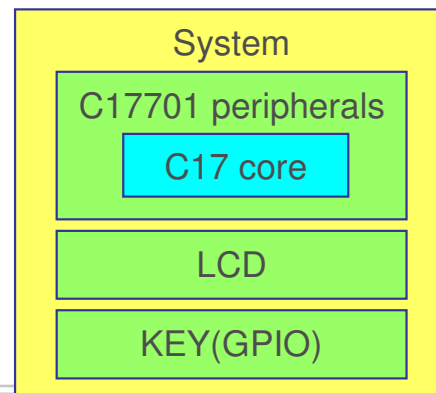
- GNU GDB debugger with insight GUI
- Same base with GNU33
- New timer break function
- Support 4 targets
 - ICDmini17
 - ICD board in SVT17701
 - Core simulator
 - Embedded System Simulator



- Toppers/osek17
 - Based on Toppers/osek which is popular as open source RTOS delivered by the Toppers project.
 - Compliant with the OSEK/VDX specification ver.2.2.1 ECC-2
 - Kernel resource requirement
 - ROM 6.7kB
 - RAM 63byte
 - Dispatch cycle around 440 to 590 cycle

GNU17v.1.0 (5) Simulator

- Core simulator
 - C17 core simulator
- Embedded System Simulator
 - Simulate C17 core
 - + C17701 peripherals
 - + LCD(dot-mtx), Key(GPIO) with GDB debug function.
 - Similar as C88/C63 Embedded System Simulator.
 - It supports more debug function than ICD with actual MCU to assist debugging.



GNU17v.1.0 (6) Sample program

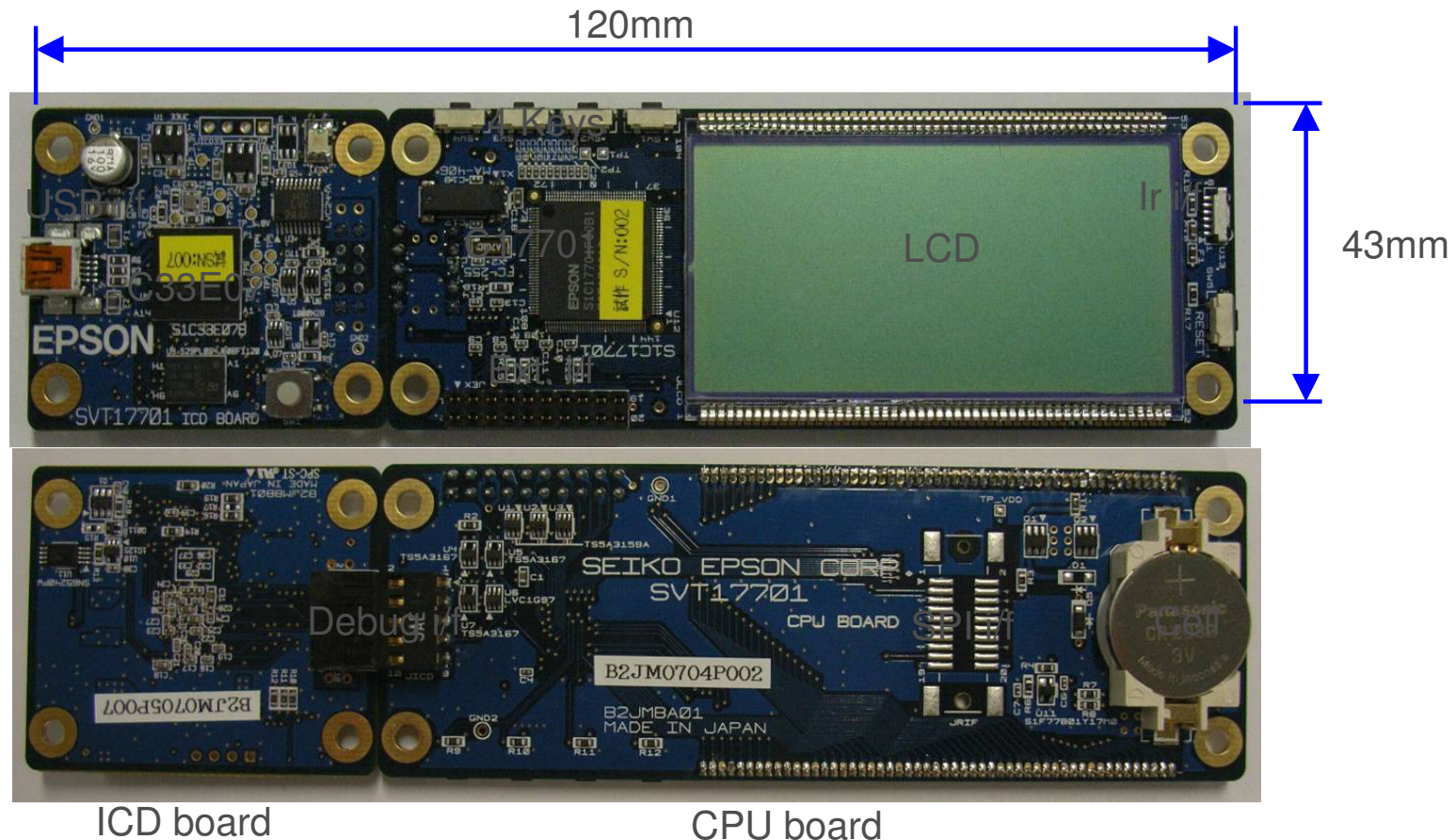
- Simple C17701 peripherals driver sample program working on the SVT17701
- Power management (Clock management, Halt/ Sleep etc.), LCD Dr., SVD, Timers, UART, SPI, ...

- Small, light weight, low price
- USB i/f, on chip debugger
- USB bus powered
- Only 4 pins connection with a target MCU (board)
- Selectable target i/f voltage level (3.3V, 1.8V, target VCC)
- Standalone flash writer function
- Controls target system reset
- Hot plugging: Connects with a target board running user program and break it, to analyze the target program status

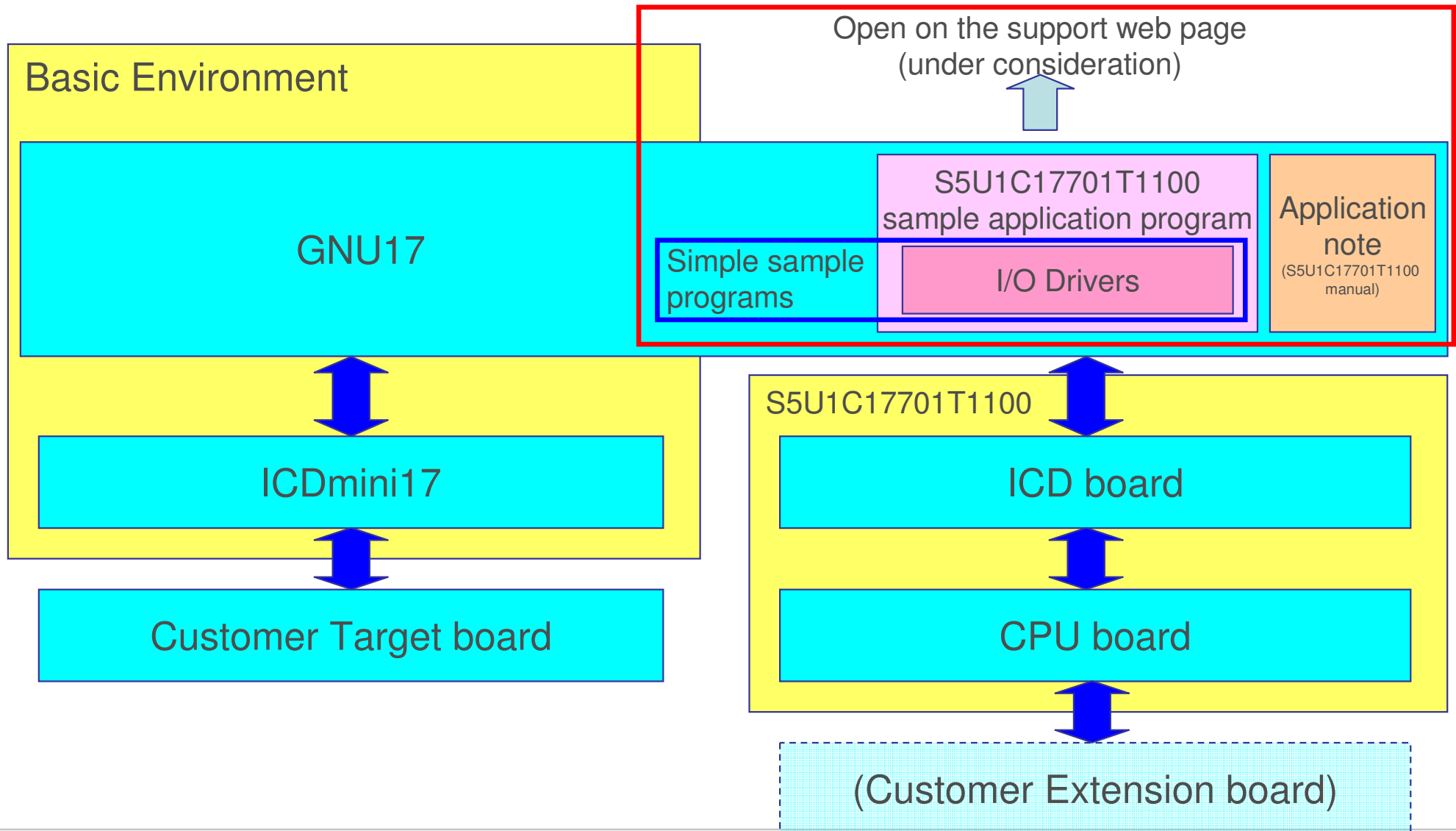


S5U1C17701T1100

- Simple, small and low cost software evaluation board for S1C17701 with
 - Simple ICD function for S5U1C17701T1100 exclusive use on the ICD board
 - S1C17701 with 32com x 56seg dot matrix B/W LCD, 4 keys, IR i/f, SPli/f, GPIO i/f on the CPU board
- Automatically selected supply: coin cell or supply from ICD board



GNU17 and S5U1C17701T1100



Comparison with C33/C88 Tools (1)

Tool	Function (Performance)	GNU33	GNU17	S1C88 C compiler package	notes
WB/IDE	platform	Eclipse	←	Original	
	Function	○	○	△ (no source code editor, etc.)	
C++ compiler		○	×	×	
C compiler	platform	GCC	←	Original (Tasking)	
	Code size ratio	○ (1.1) (-O1,-medda32)	⊙ (1.0) (-o,16bitP,shortO)	△ (1.5) (-o,small model)	(modified Dhrystone1.1)
RTOS		○ Toppers/jsp (ROS33G)	○ Toppers/osek (Toppers/jsp)	×	none
Tool	Function (Performance)	GNU33	GNU17	S1C88 C compiler package	notes
ICE/ICD	Overview	On-chip ICE	←	Full ICE	
	Target board i/f	○ 4 - 7 (10) pin 1.8V/3.3V	⊙ 4 pin 1.8V/3.3V/Target	△ MCU full pin no. 3.3V	
	customize for each model	○	○	△ (FPGA data, add-on board)	
	Flash writer function	○	○	△ (independent tool)	
	cubage, weight	○	⊙	△	
	price	○	⊙	△	

Comparison with C33/C88 Tools (2)

Tool	Function (Performance)	GNU33	GNU17	S1C88 C compiler package	notes
Debugger	platform	GNU GDB	←	Original	
	Execution function	⊙	○ (no "cycle count", "cold reset")	Δ (no "go to cursor", "auto step")	GNU17 GDB can control the target system reset via ICDmini
	Break function	○ (2 hard-PC-break, 1 hard-data-break)	Δ → ⊙ (by ES-Sim) (1 hard-PC-break, 0 hard-data-break, ...timer break is available)	⊙ (3 hard-PC-break, 4 hard-data-break, sequential break, access error break is available)	GNU17 ES-Sim has no hard-break limitaiton, and supports access error break
	Log function	○ (no coverage)	Δ → ⊙ (by ES-Sim) (no trace, coverage)	⊙	GNU17 ES-Sim has the trace function and will get the coverage function in future.
	Simulator target	Δ (no ES-Sim)	○ → ⊙ (same debugger i/f)	⊙ (Independent tool, ... various utilities)	Improve GNU17 ES-Sim utilities and functions in future.
	Other function	○	○	Δ (no termial i/o, variable display, save memory data)	

- | | |
|-------------------|----------|
| • GNU17 v.1.0 | Released |
| • ICDmini17 v.1.0 | Released |
| • S5U1C17701T1100 | 07/09/E |
| • S5U1C17801T1100 | 07/10/E |
| • S5U1C17501T1100 | 07/11/E |